Removing Bad Influence: Identifying and Pruning Detrimental Users in Collaborative Filtering Recommender Systems

Philipp Meister[1], Lukas Wegmeth[1], Tobias Vente[1] and Joeran Beel[1]

[1]Intelligent Systems Group, University of Siegen, Germany

Abstract

Recommender systems commonly employ Collaborative Filtering to generate personalized recommendations, forming an implicit social network where users influence each other's recommendations based on their preferences. In this paper, we show that it is possible to identify users with detrimental influence—those who negatively affect the recommendations of others—and that merely removing specific detrimental users from the training data can improve system performance. We apply a Leave-one-out analysis across five datasets to capture how recommendations change if a specific user is removed. Based on that data, we quantify positive and negative influences and implement a pruning strategy to remove detrimental users. Importantly, our strategy still provides recommendations to the pruned users by recommending the most popular items. We evaluate our pruning strategy on five commonly used datasets, including MovieLens, Amazon, and LastFM. We show that pruning detrimental users increases kNN performance, achieving an average performance increase of 3% for Item-Item kNN while removing 3.56% of users from the training data. Our findings highlight the potential of influence-based pruning to enhance recommender systems by increasing performance and creating resilience against detrimental influence.

Keywords

Collaborative Filtering, Recommender Systems, User Influence, User Pruning

1. Introduction

Collaborative Filtering (CF) algorithms make recommendations based on the principle that users who agree on the same items will do so in the future. The result is a system in which each user's recommendations are primarily determined by their similarity to other users. Due to users' influence on each other, Lathia et al. [1] interpret kNN CF recommender systems as implicit social networks. Much like in a social network, users' influence varies widely, resulting in a few users who significantly impact the overall system's behavior [2]. For our analysis, we define a user's influence as their ability to change other users' recommendations with their ratings.

In the quest to increase the performance and robustness of recommender systems, those influential users [3] are an important asset for recommender system engineers. For example, influential users can improve recommendations by rating new items, thereby addressing the cold start problem for those items [4]. On the other hand, bad actors, e.g., users with fake profiles or users who inject fake ratings to push certain items, could exploit the power of influential users and hurt recommendations. Wilson et al. [3] find that depending on the dataset and algorithm, using just 1% of users for such an attack results in significant performance reductions.

The potential of influential users to either enhance or erode the quality of recommendations leads us to an intriguing possibility. We hypothesize that there are real users in the training data whose inclusion negatively impacts the recommendations for other users and that removing these users can improve overall system performance. We call them detrimental users. Intuitively, removing many or all detrimental users should improve recommendations. However, it is a common assumption that less training data decreases performance [5], so removing many detrimental users may hurt rather than increase performance. To investigate whether detrimental users exist and how removing them affects recommendations, we examine the following research question:

> **RQ:** How can we identify and separate detrimental users?

To answer our research question, we analyze the influence of every individual user in five popular datasets using three CF kNN and matrix factorization algorithms to quantify user influence on ranking predictions. We show that it is possible to identify detrimental users who negatively impact the performance of other users and that pruning the most detrimental users can improve overall recommendations.

The source code reproducing all the results presented in this paper is available at our GitHub[1].

2. Related Work

Several aspects of user influence in recommender systems have been the subject of previous research. Rashid et al. [6] propose a general approach to determine user influence in rating-based recommender systems and analyze User-User and Item-Item CF systems using the Hide-one-User or Leave-one-out method. They discover correlations between user influence and several simple heuristics and use them to create a regression model to estimate user influence. The model predictions have a squared correlation coefficient of 0.94 for User-User and 0.99 for Item-Item, indicating that simple heuristics can estimate influence. Morid et al. [7] discover similar influence heuristics employing the same approach as Rashid et al. [8]. However, neither approach distinguishes between positive and detrimental influence.

In more recent work, Eskandanian et al. [2] study influential users in CF systems across different domains. Their analysis shows that the effect of influence is generally more substantial in matrix factorization systems compared to

[1]https://code.isg.beel.org/influence-pruning/

RobustRecSys: Design, Evaluation, and Deployment of Robust Recommender Systems Workshop @ RecSys 2024, 18 October, 2024, Bari, Italy.

philipp.meister@student.uni-siegen.de (P. Meister); lukas.wegmeth@uni-siegen.de (L. Wegmeth); tobias.vente@uni-siegen.de (T. Vente); joeran.beel@uni-siegen.de (J. Beel)

0009-0008-6814-9668 (P. Meister); 0000-0001-8848-9434 (L. Wegmeth); 0009-0003-8881-2379 (T. Vente); 0000-0002-4537-5573 (J. Beel)

kNN, and several factors, including centrality, number of ratings, and similarity to the average user, can identify influential users. Furthermore, they find that the effect of influence depends on the parameters like features for matrix factorization. Like Rashid et al. [6, 8], the influence discrimination model used for the underlying analysis does not account for positive or negative influence, making it impossible to identify detrimental influential users.

Wilson et al. [3] discover that it is possible to hurt CF kNN recommender systems by conducting a targeted power user attack. In contrast, Seminario et al. [9] study the same for matrix factorization CF. In this context, power users are synonymous with our definition of influential users. They find that when power users are injected with biased ratings for new items, the MAE for User-User CF rises by up to 3% on the MovieLens 1M dataset [3]. This could imply that influential users harm recommendations depending on their rating profile. Additionally, their results show that Item-Item kNN is less vulnerable to power user attacks than User-User kNN and matrix factorization.

Existing research indicates that just a few influential users have the potential to considerably change recommendations [6, 3, 9], both positively and negatively. We expand on previous research by using implicit feedback data, e.g., unweighted user interactions, and interpreting influence as a multi-dimensional metric. Furthermore, we distinguish between positive and detrimental influence and study how detrimental users can be identified and how pruning them affects recommender system performance.

3. Method

We examine detrimental users and their effect on recommendations in two parts. The first consists of a user influence analysis, which aims to identify detrimental influential users by quantifying influence via different metrics. To achieve this, we adopt the Leave-one-out (LOO) concept described by Rashid et al. [6] and developed a pipeline to capture influence data for every user. In the second part of the analysis, we use the obtained influence data to study how pruning users from the training data based on different influence metrics changes performance. One important remaining issue is that pruning removes valuable training data and disregards pruned users, which Beel et al. [10] identified as a widespread problem in recommender system research. To avoid that, we do not simply remove pruned users but instead calculate their recommendations separately, recommending the most popular items.

We use five datasets in our analysis: ML-100k, ML-1M [11], Last.FM [12], Amazon-Digital-Music and Amazon-Luxury-And-Beauty [13]. We transform explicit feedback data, e.g., ratings, into implicit feedback data, e.g., interactions, treating every rating as a positive interaction and evaluate the NDCG@10 of User-User kNN, Item-Item kNN, and Alternating Least Squares (ALS) CF on each dataset. We use the algorithm implementations of LensKit [14].

The first part of our analysis follows a simple question: if one specific user is removed from the training data, how does the NDCG@10 change for every other user? To answer this question, we implement the following LOO pipeline. First, we calculate the baseline result for each algorithm, e.g., the NDCG@10 performance of each algorithm considering all users. From the obtained results, we build a vector $\mathbf{b}$, with each entry b_i representing the baseline NDCG@10 for a user i. Then, we prune the user i from the training and test data. We train a new model and calculate the recommendations on this pruned data. These results form a vector $\mathbf{r}_i$ for each pruned user i, with $r_{i,j}$ being the NDCG@10 for user j calculated without user i in the training data. The NDCG@10 of the pruned user i is set to 0 because the user receives no recommendations. The basis of the influence analysis is the difference between the pruned results $\mathbf{r}$ and baseline results $\mathbf{b}$. For every user i, a vector $\Delta\mathbf{r}_i = \mathbf{r}_i - \mathbf{b}$ describes this difference. Pruning successively every user i results in a Matrix R with

$$R = (\Delta\mathbf{r}_i)_{i=1,\dots,n}$$

where n is the total number of users in the dataset. If for users i and j $\Delta r_{i,j} > 0$ holds, user j receives better recommendations when user i is not in the training data, ergo user i has a detrimental influence on user j. Conversely, if $\Delta r_{i,j} < 0$ holds, the existence of user i in the data improves user j's recommendations. Using R, we calculate the following four normalized influence metrics.

The ***influence mean*** μ is the difference between the baseline and pruned mean NDCG@10. It describes how the overall system performance changed compared to the baseline performance due to pruning the user i. A feature of μ_i is that it depends on user i's baseline performance because i's performance on the pruned dataset is 0. To address this issue, we introduce the ***cleaned influence mean*** γ, which removes the pruned user i from the influence mean calculations. Furthermore, we introduce the ***influence difference*** δ, which we derive from the NPD metric presented by Rashid et al. [6]. It calculates the difference between the number of users influenced positively and negatively by user i. Finally, the ***influence score*** α accounts for the cleaned influence mean and the influence difference by calculating their difference with $\alpha = \delta - \gamma$.

To test whether pruning multiple detrimental users from the training data based on the acquired influence data improves recommendations, we evaluate an optimal pruning strategy on all datasets and algorithms using user-based five-fold cross-validation. We use random search to identify the optimal pruning threshold of each influence metric for each dataset and algorithm and prune users based on this optimal pruning strategy.

4. Results & Discussion

The result of pruning detrimental users is illustrated in Figure 1. Item-Item kNN benefits the most with an average performance increase of 3% while User-User kNN also shows improvements but on a lower level with around 0.2%. ALS is, on average, negatively affected by pruning detrimental users. The relative performance change for the users remaining in the training data is, on average, around 0.5 percentage points better than all users combined. This is expected since the pruned users are recommended the most popular items, which are worse than CF.

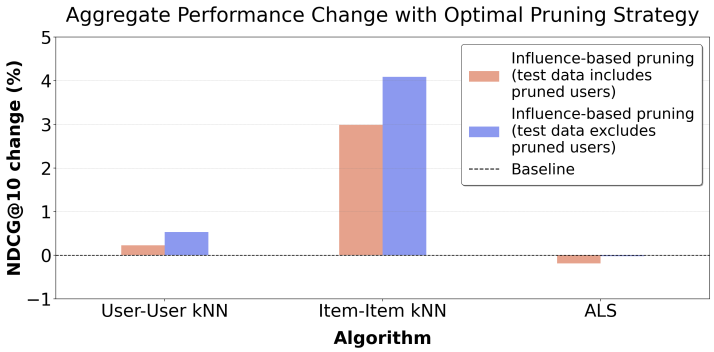

Figure 1: Aggregated performance change over all five datasets after pruning detrimental users with an optimal threshold and metric. Baseline and pruned results are calculated using five-fold cross-validation.

Algorithm	Pruned users (%)
User-User kNN	0.61%
Item-Item kNN	3.56%
ALS	0.44%

Table 1: The average percentage of users pruned from the training data for all tested algorithms.

The amount of pruned users varies significantly depending on the dataset. Table 1 shows an average, with over 3.5% of users pruned for Item-Item kNN, confirming that removing multiple detrimental users can improve recommendations despite reducing the training data. We observe that the optimal influence metric and threshold depend on the dataset and algorithm. The performance increase in our experiments varies depending on the dataset, with larger datasets benefitting more. For example, we observe a three times higher relative performance improvement for ML-1M than ML-100K.

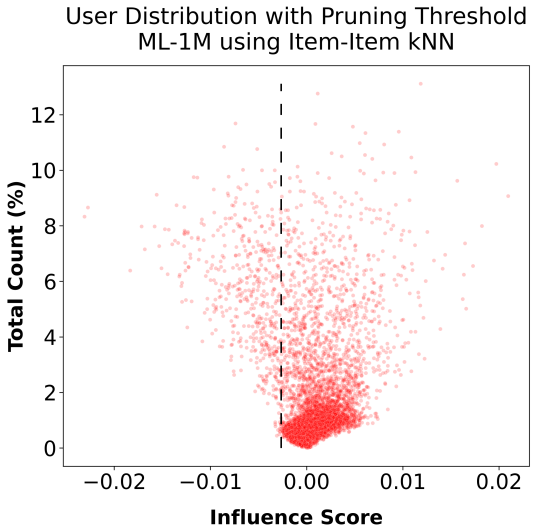

Figure 2: Correlation between the influence score of a user and the total number of other users the user influenced. The dashed line represents the threshold below which we remove all users from the training data.

To illustrate, Figure 2 shows the user distribution in the ML-1M dataset using Item-Item kNN. The dispersion of the influence score increases with rising influence. This leads to some influential users significantly negatively affecting other users. Pruning all users below the threshold shown in Figure 2 leads to a considerable overall performance improvement of over 7% for ML-1M Item-Item kNN.

The results from our pruning analysis answer our initial research question by confirming that the influence metrics we introduce can identify and differentiate users with positive and detrimental influences. The performance improvement we observe, especially for Item-Item kNN, demonstrates that the users we identify as detrimental harm the recommendations of other users. However, this effect depends on the algorithm, as shown by the reduced performance of ALS when pruning users. Future work should focus on understanding the characteristics of detrimental users and try to identify them based on heuristics without the need to conduct a computationally expensive LOO analysis.

References

[1] N. Lathia, S. Hailes, L. Capra, knn cf: a temporal social network, in: Proceedings of the 2008 ACM conference on Recommender systems, 2008, pp. 227–234.

[2] F. Eskandanian, N. Sonboli, B. Mobasher, Power of the few: Analyzing the impact of influential users in collaborative recommender systems, in: Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization, 2019, pp. 225–233.

[3] D. C. Wilson, C. E. Seminario, When power users attack: assessing impacts in collaborative recommender systems, in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 427–430.

[4] S. S. Anand, N. Griffiths, A market-based approach to address the new item problem, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 205–212.

[5] G. Adomavicius, J. Zhang, Impact of data characteristics on recommender systems performance, ACM Transactions on Management Information Systems (TMIS) 3 (2012) 1–17.

[6] A. M. Rashid, G. Karypis, J. Riedl, Influence in ratings-based recommender systems: An algorithm-independent approach, in: Proceedings of the 2005

SIAM International Conference on Data Mining, SIAM, 2005, pp. 556–560.
[7] M. A. Morid, M. Shajari, A. H. Golpayegani, Who are the most influential users in a recommender system?, in: Proceedings of the 13th international conference on electronic commerce, 2011, pp. 1–5.
[8] A. M. Rashid, Mining influence in recommender systems, 2007.
[9] C. E. Seminario, D. C. Wilson, Assessing impacts of a power user attack on a matrix factorization collaborative recommender system, in: The Twenty-Seventh International Flairs Conference, 2014.
[10] J. Beel, V. Brunel, Data pruning in recommender systems research: Best-practice or malpractice, ACM RecSys (2019).
[11] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) 5 (2015) 1–19.
[12] I. Cantador, P. Brusilovsky, T. Kuflik, 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011), in: Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, ACM, New York, NY, USA, 2011.
[13] J. Ni, J. Li, J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2019, pp. 188–197.
[14] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: Proceedings of the 29th ACM international conference on information & knowledge management, 2020, pp. 2999–3006.